(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 0 798 636 B1

(12) EUROPEAN PATENT SPECIFICATION

(45) Date of publication and mention of the grant of the patent:
05.06.2002 Bulletin 2002/23

(51) Int Cl.[7]: **G06F 9/45**

(21) Application number: **97103331.1**

(22) Date of filing: **28.02.1997**

(54) Method, system and computer program product for register allocation using multiple interference graphs

Verfahren, System und Rechnerprogrammprodukt zur Registerzuweisung unter Verwendung mehrerer Interferenzgraphen

Procédé, système et produit logiciel informatique d'allocation des registres en utilisant plusieurs graphes d'interférence

(84) Designated Contracting States:
DE FR GB IT NL

(30) Priority: **25.03.1996 US 621409**

(43) Date of publication of application:
01.10.1997 Bulletin 1997/40

(73) Proprietor: **SUN MICROSYSTEMS, INC.**
Mountain View, California 94043-1100 (US)

(72) Inventor: **Goebel, Kurt J.**
Mountain View, California 94043 (US)

(74) Representative: **Kahler, Kurt, Dipl.-Ing.**
Patentanwälte
Kahler, Käck, Fiener et col.,
Vorderer Anger 268
86899 Landsberg/Lech (DE)

(56) References cited:
EP-A- 0 051 131

- **ACM TRANSACTIONS ON PROGRAMMING LANGUAGES, vol. 12, no. 4, October 1990, NEW YORK, NY, US, pages 501-536, XP000255876 F. C. CHOW ET AL: "The Priority-Based Coloring Approach to Register Allocation"**
- **COMPUTER LANGUAGES, vol. 6, 1981, pages 47-57, XP000255875 G. J. CHAITIN ET AL: "REGISTER ALLOCATION VIA COLORING"**

Note: Within nine months from the publication of the mention of the grant of the European patent, any person may give notice to the European Patent Office of opposition to the European patent granted. Notice of opposition shall be filed in a written reasoned statement. It shall not be deemed to have been filed until the opposition fee has been paid. (Art. 99(1) European Patent Convention).

EP 0 798 636 B1

Printed by Jouve, 75001 PARIS (FR)

Description

BACKGROUND OF THE INVENTION

[0001] This invention relates to the optimization of register allocation in data processing systems during program compile time. More particularly, this invention relates to register allocation optimization using interference graph coloring techniques.

[0002] Register allocation using interference graph coloring techniques is known and has been widely adopted for use in assigning real registers to symbolic or virtual registers when a program is being compiled prior to execution. The basic reference in this field is "Register Allocation Via Coloring", G. J. CHAITIN ET AL., Computer Languages Vol 6 pages 47-57, 1981. fax

[0003] In general, the technique proceeds by creating in the data processing system compiler a register interference graph using symbolic or virtual registers, with the symbolic registers as individual nodes in the graph. For those nodes in which the contents are concurrently live, such nodes are connected by an edge. Once the interference graph is constructed, those nodes having a degree less than k, where k is the number of real registers available in the data processing system, are deleted from the graph in some logical fashion. Nodes of degree greater than or equal to k are selected for a spilling operation, in which the contents of the real registers (when the program is executed) will be stored elsewhere for later recall. A new interference graph is thereafter constructed, and each node is examined in the same fashion. This iterative process continues until all nodes have been removed from the interference graph. Thereafter, the nodes are examined in reverse order and assigned individual colors (in actuality real register numbers) and the process is thus completed.

[0004] While generally effective in allocating real registers when a program is being compiled, many specific applications provide conditional constraints which result in less than optimal register allocation as a result of the interference graph coloring technique. For example, in some data processing systems employing single precision floating point operations, real registers are bound together as pairs, with a resulting inability of the data processing system to distinguish between individual registers of a pair. If one of the paired registers is allocated to a first node and the other one of the register pair is allocated to a second node with common latency with the first node (i.e., the first and second nodes are linked with an edge), an interlock condition will exist in which the system must wait several machine cycles for the completion of the execution of a given instruction before the next instruction is permitted to proceed. While the delay in execution introduced by a float interlock condition is usually not so great as that induced by a spill operation, the execution time of that portion of the program still suffers, which is undesirable and less than optimal.

SUMMARY OF THE INVENTION

[0005] The invention comprises an improved interference graph register allocation technique which avoids the execution time delay introduced by conditional constraints (such as the single precision floating point register pairing constraint) and provides optimal allocation of real registers during compilation of the program into machine executable form.

[0006] From a process standpoint, the invention comprises a method of optimizing the allocation of real registers in a data processing system CPU which proceeds by creating a primary interference graph having nodes representing variable registers and primary edges linking nodes with concurrent latency, the number of edges of a given node being termed the degree of the node. A secondary interference graph is then created having nodes representing virtual registers and secondary edges linking nodes with conditional conflicts which can be tolerated but which are not optimal, such as a node requiring the use of one of a pair of paired registers in a single precision floating point operation. Nodes having degrees less than the number of available CPU real registers are selected from the primary interference graph until all nodes have been so selected. Thereafter, an attempt is made to allocate real registers to the selected nodes in the reverse order of their selection by first determining whether a register can be allocated for a selected node using the edge constraints of both the primary and secondary interference graphs. If so, allocation of the real register to a given node is done on the basis of both the primary and secondary interference graphs. If a register cannot be allocated for a selected node using the edges of both the primary and secondary interference graphs, the real register is allocated on the basis of the primary graph alone. The step of allocating is preferably performed on nodes in the reverse order from which the nodes were originally selected: i.e., the first selected node receives the last register allocation.

[0007] From a system standpoint, the invention comprises a data processing system having a CPU with a fixed number of allocatable registers, and a compiler including a first procedure for creating a primary interference graph having nodes representing virtual registers and primary edges linking nodes with concurrent latency, the number of edges of a given node being the degree of the node, a second procedure for creating a second interference graph having nodes representing virtual registers and secondary edges linking nodes with conditional conflicts, a third procedure for selecting nodes from the primary interference graph having a degree less than the number of CPU registers

until all nodes have been selected from the primary interference graph, and a fourth procedure for allocating real registers to the selected nodes by first determining whether a register can be allocated for a selected node using the edges of both the primary and secondary interference graphs and, if so, allocating a real register on that basis; and, if not, allocating a real register using the edges of the primary graph alone. The allocating procedure is performed on nodes in the reverse order from which the nodes were originally selected. In a specific embodiment of the procedures, the secondary interference graph edges link a virtual register requiring use of a paired register with another virtual register having common latency.

[0008] The invention also comprises a computer program product as set out in claim 7.

[0009] For a fuller understanding of the nature and advantages of the invention, reference should be made to the ensuing detailed description, taken in conjunction with the accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

[0010]

FIG. 1 is an interference graph illustrating the invention;
FIG. 2 is a flow chart illustrating the overall interference graph coloring technique of the invention;
FIG. 3 is a flow chart illustrating the use of a primary and a secondary interference chart according to the invention; and
FIG. 4 is a block diagram of a data processing system incorporating the invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

[0011] Turning now to the drawings, Fig. 1 is a simplified interference graph illustrating the invention. As seen in this figure, the interference graph has three nodes: node 12 representing virtual register 100, node 14 representing virtual register 101 and node 16 representing virtual register 102. Nodes 12 and 14 are joined by a first primary edge 13 denoting common latency between the contents of these two registers. Similarly, nodes 12 and 16 are joined by a primary edge 15 illustrating common latency between the contents of these two registers. In addition, nodes 12 and 14 are joined by a secondary edge 17 denoting a secondary conflict between the contents of these two nodes. The secondary conflict is a conditional conflict: i.e., one which can be tolerated but which, if avoided, will optimize the assignment of real registers to the virtual registers at nodes 12 and 14. For example, in the single precision floating point operation noted above, the secondary link 17 may denote that the contents of one of the nodes 12, 14 is stored in one of two paired registers which cannot be readily distinguished by the system. In such a case, the real register assignment for one of the two nodes 12, 14 should be some other register than the remaining register in the pair.

[0012] Interference graph 10 of Fig. 1 is actually constructed during compile time by assigning a pair of vectors to each node: one vector representing primary links and the other vector representing the secondary links. Although the interference graph of Fig. 1 illustrates only three nodes for simplicity and clarity, it is understood that many nodes typically exist in a given interference graph.

[0013] Once the graph has been constructed, real register assignments or "coloring" is attempted using both the primary and secondary components of the interference graph. Fig. 2 is a flow chart illustrating the overall process. As seen in this figure, the register allocation technique of the present invention begins with a step of building the primary interference graph and the secondary interference graph in step 22. Next, a check is made to determine whether there is a node in the primary interference graph only having a degree less than the number of machine registers. If not, then a conventional spill routine 25 is entered, and a new primary and secondary interference graph are constructed in a second pass through step 22.

[0014] If there is a node with a degree less than the number of machine registers, then that node and its edges are marked for selection in step 26. Next, the graph is tested in step 28 to determine whether all nodes have been selected and, if not, the routine returns to step 24 to select another node and its edges.

[0015] Once all of the nodes from the primary interference graph have been selected, real register numbers are assigned to the individual selected nodes in step 30, in reverse order from the order of selection of the nodes and using both the primary and secondary interference graphs. Lastly, the program is rewritten in step 32 by replacing symbolic registers with real machine register numbers, which terminates the register allocation technique.

[0016] Fig. 3 illustrates step 30 of assigning real register numbers in more detail. As seen in this figure, as each selected node is visited, a test is made in step 42 to determine whether a real register can be allocated for this node in a manner which satisfies both the primary and secondary interference graphs. If so, the register number is allocated using both the primary and secondary interference graphs in step 44. If not, a register number is allocated using the primary interference graph alone in step 46. Thereafter, a test is made to determine whether the last node has been allocated a real register. If not, the routine returns to step 42. Once the last node has been allocated a real register,

the routine enters the last step 32 of the overall allocation process.

[0017] The register allocation technique described above is implemented in a data processing system such as that illustrated in Fig. 4. As seen in this figure, a program in a high level form (such as source code written in C or C++) is supplied from a source 50 to a compiler 52. The compiler 52 conducts the register allocation procedure (and other optimization procedures) and produces assembly code which is supplied to a CPU 54 having main memory 56, mass storage 57 and various I/O devices 58. The CPU contains the real registers which are allocated to the virtual or symbolic registers as a result of the interference graph coloring technique. As noted above, the secondary interference graph is constructed in order to indicate conflicts between nodes which are conditional and which thus may be tolerated albeit at the expense of optimal use of the real registers in executing various program segments. The following is an example of a Sparc assembly language instruction sequence in which the secondary interference graph edges have been used (in combination with the primary interference graph edges) in order to allocate registers in a manner to eliminate the conditional conflict of a floating point interlock.

```
*****************************************************************************
****
/* 0x0208          51 (80 86) */          fdivs   %f7,%f9,%f2
/* 0x020c          45 (80 82) */          ld      [%sp+156),%10
/* 0x0210           0 (80 80) */          add     %14,%lo(_BLNK__+1050620),%12
*****************************************************************************
****
/* 0x0214          44 (81 82) */          st      %o0,[%sp+100]
/* 0x0218           0 (81 81) */          add     %16,%lo(_BLNK__+6314000),%i1
*****************************************************************************
****
/* 0x021c          56 (82 84) */          ld      [%i5],%f0
/* 0x0220          45 (82 82) */          add     %10,1,%o0
/* 0x0224           0 (82 82) */          sethi   %hi(_BLNK__+7366676),%11
*****************************************************************************
****
/* 0x0228          47 (83 85) */          ld      [%i0-8],%f10
/* 0x022c          56 (83 83) */          cmp     %o0,1
/* 0x0230          55 (83 83) */          or      %g0,%o0,%16
*****************************************************************************
****
/* 0x0234          45 (84 85) */          st      %o0,[%sp+96]
/* 0x0238           0 (84 84) */          add     %11,%lo(_BLNK__+7366676),%14
*****************************************************************************
****
/* 0x023c          56 (85 87) */          ld      [%sp+140],%f4
*****************************************************************************
****
```

[0018] In this example, instruction groups which issue in the same machine cycle are separated by the lines constructed of asterisks. The first instruction is an instruction termed the fdivs instruction. The pair of numbers (80 86) indicates that the instruction will issue in cycle 80 and is expected to complete in cycle 86. Consequently, the fdivs instruction will be executing through all of the shown instruction groups (since the last group is issuing in cycle 85). The fdivs instruction is an instruction involving potential flow interlock, and since the result of the instruction is to be stored in register f2, optimal register allocation requires that no instruction use paired register f3 during any of machine cycles 80-86. Otherwise, such an instruction would be interlocked from commencing execution until the end of execution of the fdivs instruction (i.e., machine cycle 86). By selecting a real register other than f3 (and f2), the float interlock problem is avoided and the set of instructions can proceed with maximum execution speed.

[0019] While the above provides a full and complete disclosure of the preferred embodiments of the invention, various modifications, alternate constructions and equivalents may be employed. For example, while the secondary interference graph has been expressly described with reference to the single precision floating point interlock problem involving paired registers, other conditional constraints may be employed, as desired, in constructing the secondary interference graph. Similarly, the invention is not limited to the use of only a single primary and a single secondary interference graph, but may also be extended to tertiary graphs and graphs of other levels as well. Therefore, the above should not be construed as limiting the invention, which is defined by the appended claims.

Claims

1. A computer implemented method for use in a compiler (52) for allocating real registers in a CPU (24) to instructions of a target computer program which are to be executed on the CPU, said method comprising the steps of:

   creating a representation of a primary interference graph (22) having nodes representing virtual registers and primary edges linking nodes with concurrent latency, the number of edges of a given node being the degree of the node;
   creating a representation of a second interference graph (22) having nodes representing virtual registers and secondary edges linking nodes with conditional conflicts;
   selecting nodes (26) from the primary interference graph having degrees less than the number of CPU (24) registers until all nodes have been selected (28) ; and
   allocating real registers to said selected nodes by first determining whether a register can be allocated for a selected node using the edges of both the primary and secondary interference graphs (42) and, if so, allocating a real register on that basis (44); and, if not, allocating a real register using the edges of the primary graph alone (46).

2. The method of claim 1 wherein the step of allocating is performed on nodes in the reverse order from which the nodes were originally selected.

3. The method of claim 1 wherein the secondary interference graph edges link a virtual register requiring use of a paired register with another virtual register

4. A computer system comprising:

   a central processing unit or CPU (54) having a finite number of allocatable registers; and
   a compiler including a first procedure for creating a representation of a primary interference graph having nodes representing virtual registers and primary edges linking nodes with concurrent latency, the number of edges of a given node being the degree of the node, a second procedure for creating a representation of a secondary interference graph having nodes representing virtual registers and secondary edges linking nodes with conditional conflicts; a third procedure for selecting nodes from the primary interference graph having a degree less than the number of CPU registers until all nodes have been selected from the primary interference graph; and a fourth procedure for allocating real registers to the selected nodes by first determining whether a register can be allocated for a selected node using the edges of both the primary and secondary interference graphs and, if so, allocating a real register on that basis; and, if not, allocating a real register using the edges of the primary graph alone.

5. The computer system of claim 4 wherein said fourth procedure is performed on nodes in the reverse order from which the nodes were originally selected.

6. The computer system of claim 4 wherein the secondary interference graph edges link a virtual register requiring use of a paired register with another virtual register.

7. A computer program product comprising:

   a computer usable medium having computer readable code embodied therein for optimizing allocation of real registers in a computer system, the computer program product comprising:
   a first set of computer readable program code devices for creating a representation of a primary interference graph having nodes representing virtual registers and primary edges linking nodes with concurrent latency, the number of edges of a given node being the degree of the node, and for creating a representation of a secondary interference graph having nodes representing virtual registers and secondary edges linking nodes with conditional conflicts;
   a second set of computer readable program code devices configured to cause a computer to select nodes from the primary interference graph having degrees less than the number of real registers in the computer system until all nodes have been selected; and
   a third set of computer readable program code devices configured to cause a computer to allocate real registers to the selected nodes by first determining whether a real register can be allocated for a selected node using the edges of both the primary and secondary interference graphs and, if so, allocating a real register on that

basis; and, if not, allocating a real register using the edges of the primary graph alone.

8. The computer program product of claim 7 wherein said first set of computer readable program code devices is configured to cause a computer to create a representation of a secondary interference graph having secondary edges linking a virtual register requiring use of a paired register with another virtual register.

9. The computer program product of claim 7 wherein said third set of computer readable program code devices is configured to cause a computer to perform the allocation of real registers in the reverse order from which the nodes were originally selected.

Patentansprüche

1. Computerimplementiertes Verfahren zur Verwendung in einem Kompilierer (52), um reale Register in einer CPU (24) Befehlen eines Zielcomputerprogramms zuzuweisen, welche in der CPU ausgeführt werden sollen, wobei das Verfahren die Schritte umfaßt:

   Erzeugen einer Darstellung einer primären Interferenzkurve (22) mit Knoten, die virtuelle Register darstellen, und primären Kanten, die Knoten mit gleichzeitiger Wartezeit verbinden, wobei die Anzahl von Kanten eines gegebenen Knotens der Grad des Knotens ist;
   Erzeugen einer Darstellung einer sekundären Interferenzkurve (22) mit Knoten, die virtuelle Register darstellen, und sekundären Kanten, die Knoten mit bedingten Konflikten verbinden;
   Auswählen von Knoten (26) aus der primären Interferenzkurve mit Graden, die geringer sind als die Anzahl von Registern der CPU (24), bis alle Knoten ausgewählt wurden (28); und
   Zuweisen von realen Registern den ausgewählten Knoten durch zuerst Feststellen, ob ein Register für einen ausgewählten Knoten zugewiesen werden kann, unter Verwendung der Kanten von sowohl der primären als auch der sekundären Interferenzkurve (42), und wenn ja, Zuweisen eines realen Registers auf dieser Basis (44); und wenn nicht, Zuweisen eines realen Registers unter Verwendung der Kanten der primären Kurve allein (46).

2. Verfahren nach Anspruch 1, wobei der Schritt des Zuweisens an den Knoten in umgekehrter Reihenfolge, in der die Knoten ursprünglich ausgewählt wurden, durchgeführt wird.

3. Verfahren nach Anspruch 1, wobei die Kanten der sekundären Interferenzkurve ein virtuelles Register, das die Verwendung eines gepaarten Registers erfordert, mit einem weiteren virtuellen Register verknüpfen.

4. Computersystem mit:

   einer Zentralverarbeitungseinheit (CPU) (54) mit einer endlichen Anzahl von zuweisbaren Registern; und
   einem Kompilierer mit einer ersten Prozedur zum Erzeugen einer Darstellung einer primären Interferenzkurve mit Knoten, die virtuelle Register darstellen, und primären Kanten, die Knoten mit gleichzeitiger Wartezeit verbinden, wobei die Anzahl von Kanten eines gegebenen Knotens der Grad des Knotens ist, einer zweiten Prozedur zum Erzeugen einer Darstellung einer sekundären Interferenzkurve mit Knoten, die virtuelle Register darstellen, und sekundären Kanten, die Knoten mit bedingten Konflikten verbinden; einer dritten Prozedur zum Auswählen von Knoten aus der primären Interferenzkurve mit einem Grad, der geringer ist als die Anzahl von CPU-Registern, bis alle Knoten aus der primären Interferenzkurve ausgewählt wurden; und einer vierten Prozedur zum Zuweisen von realen Registern den ausgewählten Knoten durch zuerst Feststellen, ob ein Register für einen ausgewählten Knoten zugewiesen werden kann, unter Verwendung der Kanten von sowohl der primären als auch der sekundären Interferenzkurve, und wenn ja, Zuweisen eines realen Registers auf dieser Basis; und wenn nicht, Zuweisen eines realen Registers unter Verwendung der Kanten der primären Kurve allein.

5. Computersystem nach Anspruch 4, wobei die vierte Prozedur an den Knoten in umgekehrter Reihenfolge, in der die Knoten ursprünglich ausgewählt wurden, durchgeführt wird.

6. Computersystem nach Anspruch 4, wobei die Kanten der sekundären Interferenzkurve ein virtuelles Register, das die Verwendung eines gepaarten Registers erfordert, mit einem weiteren virtuellen Register verknüpfen.

7. Computerprogrammprodukt mit:

einem vom Computer verwendbaren Medium mit einem darin verkörperten maschinenlesbaren Code zum Optimieren der Zuweisung von realen Registern in einem Computersystem, wobei das Computerprogrammprodukt umfaßt:

einen ersten Satz von Vorrichtungen eines maschinenlesbaren Programmcodes zum Erzeugen einer Darstellung einer primären Interferenzkurve mit Knoten, die virtuelle Register darstellen, und primären Kanten, die Knoten mit gleichzeitiger Wartezeit verbinden, wobei die Anzahl von Kanten eines gegebenen Knotens der Grad des Knotens ist, und zum Erzeugen einer Darstellung einer sekundären Interferenzkurve mit Knoten, die virtuelle Register darstellen, und sekundären Kanten, die Knoten mit bedingten Konflikten verbinden;
einen zweiten Satz von Vorrichtungen eines maschinenlesbaren Programmcodes, die dazu ausgelegt sind, einen Computer zu veranlassen, Knoten aus der primären Interferenzkurve mit Graden, die geringer sind als die Anzahl von realen Registern in dem Computersystem, auszuwählen, bis alle Knoten ausgewählt wurden; und
einen dritten Satz von Vorrichtungen eines maschinenlesbaren Programmcodes, die dazu ausgelegt sind, einen Computer zu veranlassen, reale Register den ausgewählten Knoten zuzuweisen, durch zuerst Feststellen, ob ein reales Register für einen ausgewählten Knoten zugewiesen werden kann, unter Verwendung der Kanten von sowohl der primären als auch der sekundären Interferenzkurve, und wenn ja, Zuweisen eines realen Registers auf dieser Basis; und wenn nicht, Zuweisen eines realen Registers unter Verwendung der Kanten der primären Kurve allein.

8. Computerprogrammprodukt nach Anspruch 7, wobei der erste Satz von Vorrichtungen eines maschinenlesbaren Programmcodes dazu ausgelegt ist, einen Computer zu veranlassen, eine Darstellung einer sekundären Interferenzkurve mit sekundären Kanten zu erzeugen, die ein virtuelles Register, das die Verwendung eines gepaarten Registers erfordert, mit einem weiteren virtuellen Register verknüpfen.

9. Computerprogrammprodukt nach Anspruch 7, wobei der dritte Satz von Vorrichtungen eines maschinenlesbaren Programmcodes dazu ausgelegt ist, einen Computer zu veranlassen, die Zuweisung von realen Registern in der umgekehrten Reihenfolge, in der die Knoten ursprünglich ausgewählt wurden, durchzuführen.

Revendications

1. Procédé mis en oeuvre par un ordinateur en vue d'une utilisation dans un compilateur (52) destiné à allouer des registres réels dans une unité centrale à des instructions d'un programme informatique cible qui doivent être exécutées sur l'unité centrale, ledit procédé comprenant les étapes consistant à :

créer une représentation d'un graphe d'interférence primaire (22) comportant des noeuds présentant des registres virtuels et des arêtes primaires reliant des noeuds présentant un temps de latence simultané, le nombre des arêtes d'un noeud donné étant le degré du noeud,
créer une représentation d'un graphe d'interférence secondaire (22) comportant des noeuds représentant des registres virtuels et des arêtes secondaires reliant des noeuds présentant des conflits conditionnels,
sélectionner des noeuds (26) à partir du graphe d'interférence primaire présentant des degrés inférieurs au nombre des registres de l'unité centrale (24) jusqu'à ce que tous les noeuds aient été sélectionnés (28), et allouer des registres réels auxdits noeuds sélectionnés en déterminant tout d'abord si un registre peut être alloué pour un noeud sélectionné en utilisant les arêtes à la fois pour les graphes d'interférence primaire et secondaire (42) et, si c'est le cas, allouer un registre réel sur cette base (44), et si ce n'est pas le cas, allouer un registre réel en utilisant les arêtes du graphe primaire seul.

2. Procédé selon la revendication 1, dans lequel l'étape d'allocation est exécutée sur des noeuds dans l'ordre inverse par rapport à celui selon lequel les noeuds ont été sélectionnés à l'origine.

3. Procédé selon la revendication 1, dans lequel les arêtes du graphe d'interférence secondaire relient un registre virtuel nécessitant l'utilisation d'un registre apparié avec un autre registre virtuel.

4. Système informatique comprenant :

une unité centrale de traitement (UC ; 54) présentant un nombre fini de registres pouvant être alloués, et un compilateur comprenant une première procédure destinée à créer une représentation d'un graphe d'interférence primaire comportant des noeuds représentant des registres virtuels et des arêtes primaires reliant des noeuds présentant un temps de latence simultané, le nombre des arêtes d'un noeud donné étant le degré du noeud, une seconde procédure destinée à créer une représentation d'un graphe d'interférence secondaire comportant des noeuds représentant des registres virtuels et des arêtes secondaires reliant des noeuds présentant des conflits conditionnels, une troisième procédure destinée à sélectionner des noeuds à partir du graphe d'interférence primaire présentant un degré inférieur au nombre des registres de l'unité centrale jusqu'à ce que tous les noeuds aient été sélectionnés à partir du graphe d'interférence primaire, et une quatrième procédure destinée à allouer des registres réels aux noeuds sélectionnés en déterminant tout d'abord si un registre peut être alloué pour un noeud sélectionné en utilisant les arêtes à la fois des graphes d'interférence primaire et secondaire et, si c'est le cas, en allouant un registre réel sur cette base, et si ce n'est pas le cas, en allouant un registre réel en utilisant les arêtes du graphe primaire seul.

5. Système informatique selon la revendication 4, dans lequel ladite quatrième procédure est exécutée sur des noeuds dans l'ordre inverse par rapport à celui suivant lequel les noeuds ont été sélectionnés à l'origine.

6. Système informatique selon la revendication 4, dans lequel les arêtes du graphe d'interférence secondaire relient un registre virtuel nécessitant l'utilisation d'un registre apparié avec un autre registre virtuel.

7. Produit de programme informatique comprenant :

un support utilisable par un ordinateur comportant un code lisible par un ordinateur incorporé dans celui-ci en vue d'optimiser l'allocation des registres réels dans un système informatique, le produit de programme informatique comprenant :

un premier ensemble de dispositifs de code de programme lisible par un ordinateur destiné à créer une représentation d'un graphe d'interférence primaire comportant des noeuds représentant des registres virtuels et des arêtes primaires reliant des noeuds présentant un temps de latence simultané, le nombre des arêtes d'un noeud donné étant le degré du noeud, et destiné à créer une représentation d'un graphe d'interférence secondaire comportant des noeuds représentant des registres virtuels et des arêtes secondaires reliant des noeuds présentant des conflits conditionnels,
un second ensemble de dispositifs de code de programme lisible par un ordinateur configuré pour amener un ordinateur à sélectionner des noeuds à partir du graphe d'interférence primaire, présentant des degrés inférieurs au nombre des registres dans le système informatique jusqu'à ce que tous les noeuds aient été sélectionnés, et
un troisième ensemble de dispositifs de code de programme lisible par un ordinateur configuré pour amener un ordinateur à allouer des registres réels aux noeuds sélectionnés en déterminant tout d'abord si un registre réel peut être alloué pour un noeud sélectionné en utilisant les arêtes à la fois des graphes d'interférence primaire et secondaire et, si c'est le cas, en allouant un registre réel sur cette base, et si ce n'est pas le cas, en allouant un registre réel en utilisant les arêtes du graphe primaire seul.

8. Produit de programme informatique selon la revendication 7, dans lequel ledit premier ensemble de dispositifs de code de programme lisible par un ordinateur est configuré pour amener un ordinateur à créer une représentation d'un graphe d'interférence secondaire comportant des arêtes secondaires reliant un registre virtuel nécessitant l'utilisation d'un registre apparié avec un autre registre virtuel.

9. Produit de programme informatique selon la revendication 7, dans lequel ledit troisième ensemble de dispositifs de code de programme lisible par un ordinateur est configuré pour amener un ordinateur à exécuter l'allocation des registres réels dans l'ordre inverse par rapport à celui selon lequel les noeuds ont été sélectionnés à l'origine.

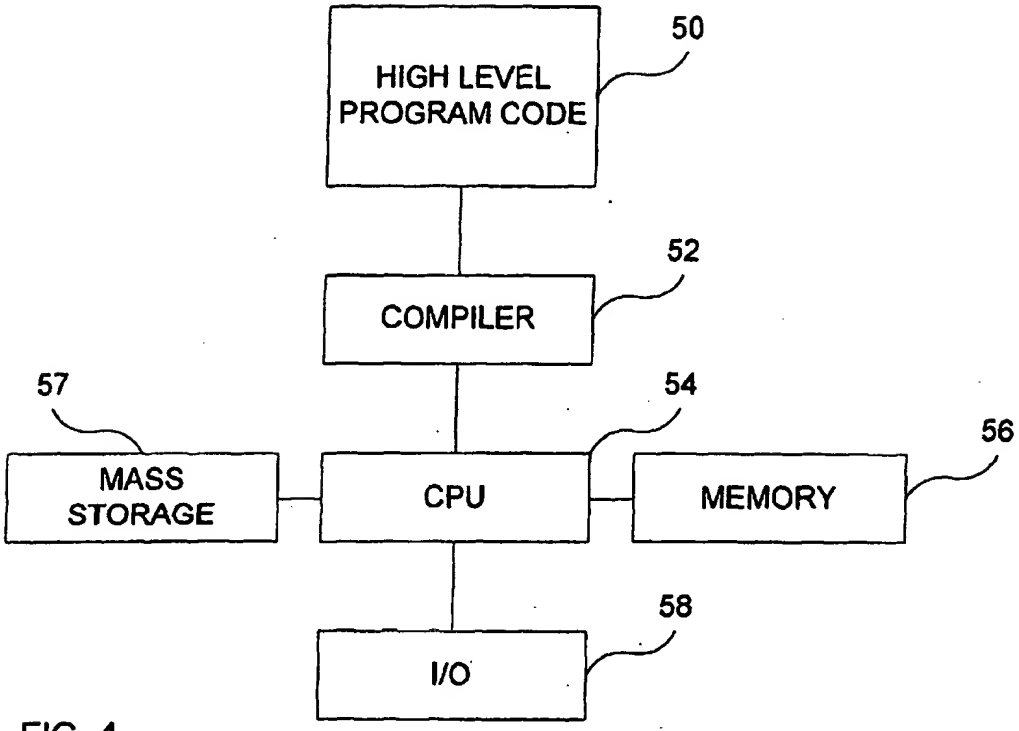
HIGH LEVEL PROGRAM CODE
50
COMPILER
52
57
54
56
MASS STORAGE
CPU
MEMORY
58
I/O

FIG. 4

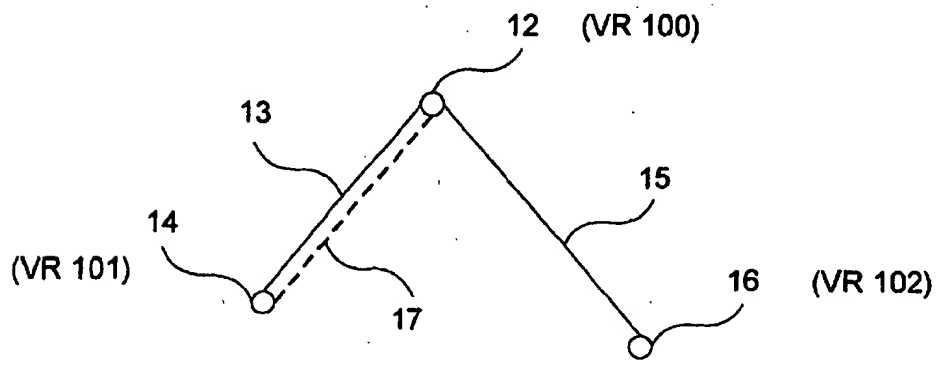
12
(VR 100)
13
15
14
(VR 101)
16
(VR 102)
17

FIG. 1

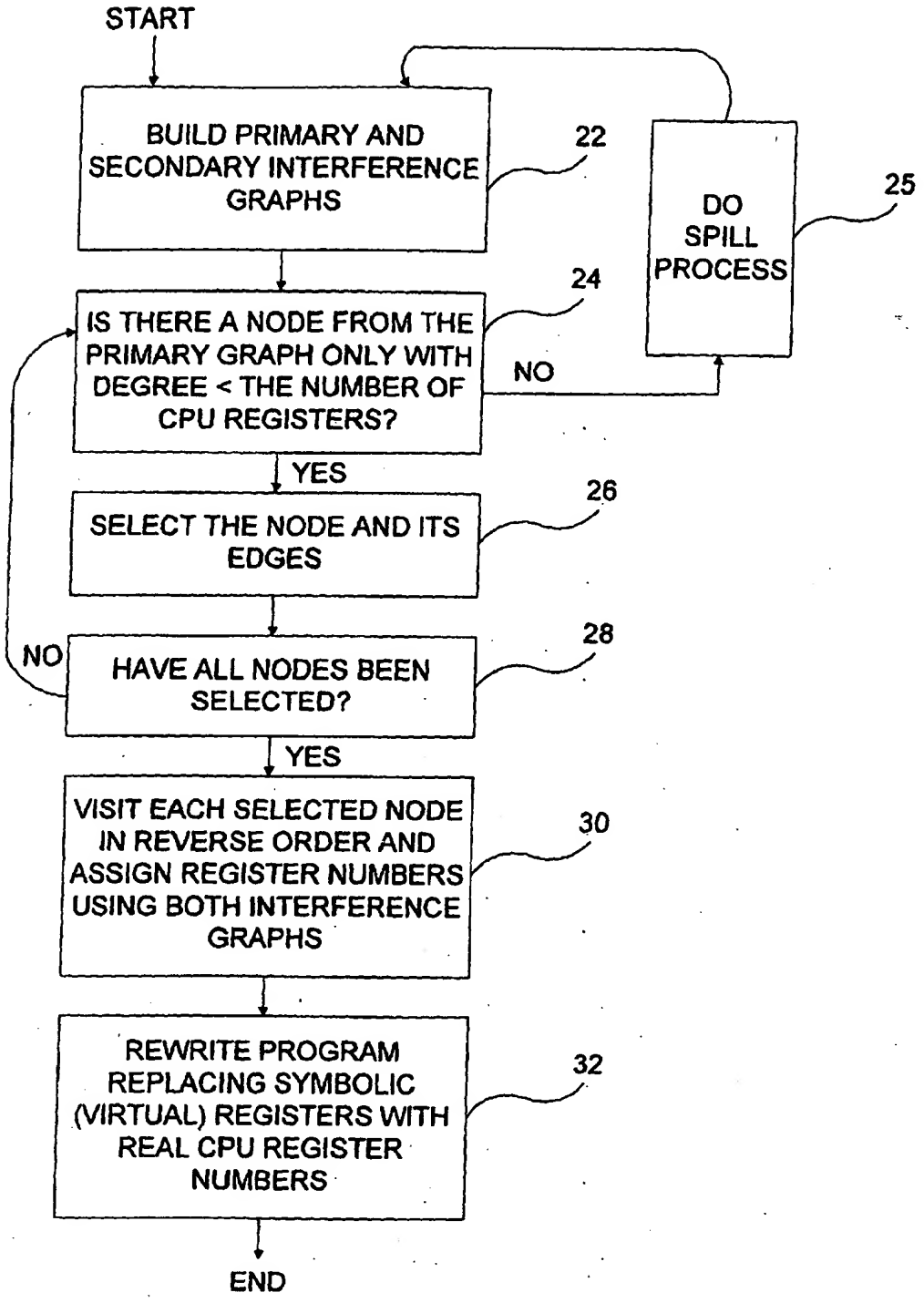
START
BUILD PRIMARY AND SECONDARY INTERFERENCE GRAPHS
22
DO SPILL PROCESS
25
IS THERE A NODE FROM THE PRIMARY GRAPH ONLY WITH DEGREE < THE NUMBER OF CPU REGISTERS?
24
NO
YES
SELECT THE NODE AND ITS EDGES
26
NO
HAVE ALL NODES BEEN SELECTED?
28
YES
VISIT EACH SELECTED NODE IN REVERSE ORDER AND ASSIGN REGISTER NUMBERS USING BOTH INTERFERENCE GRAPHS
30
REWRITE PROGRAM REPLACING SYMBOLIC (VIRTUAL) REGISTERS WITH REAL CPU REGISTER NUMBERS
32
END

FIG. 2

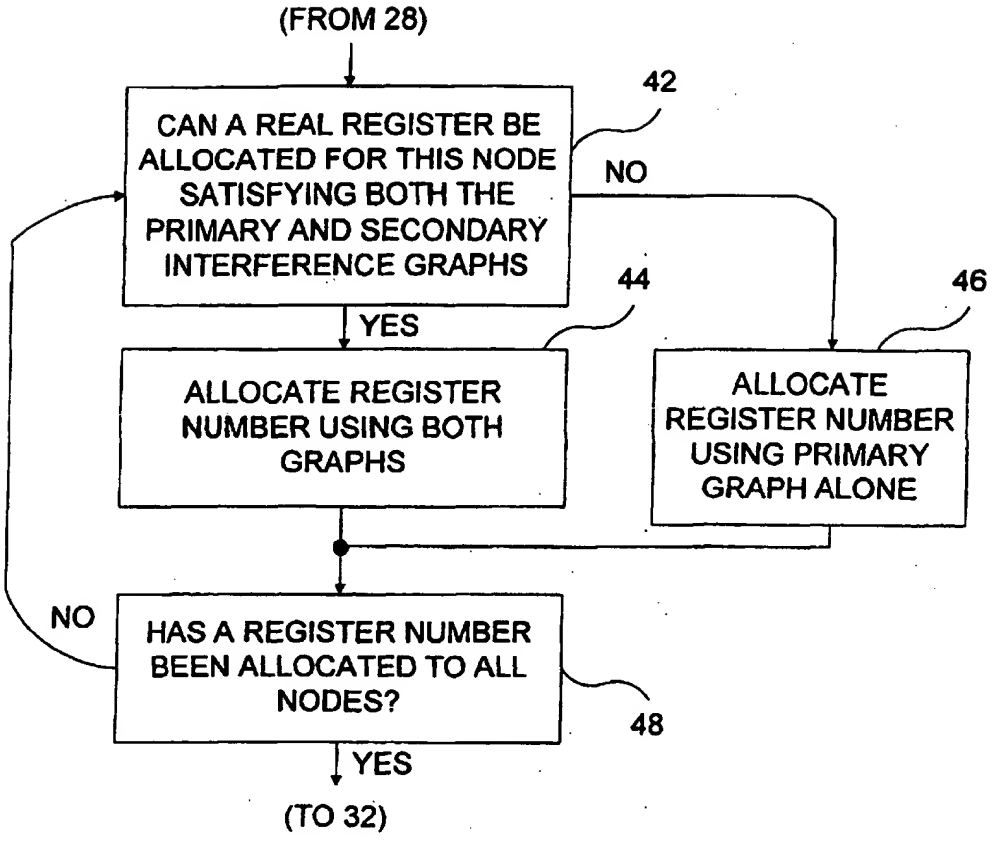
(FROM 28)
CAN A REAL REGISTER BE ALLOCATED FOR THIS NODE SATISFYING BOTH THE PRIMARY AND SECONDARY INTERFERENCE GRAPHS
42
NO
YES
44
ALLOCATE REGISTER NUMBER USING BOTH GRAPHS
46
ALLOCATE REGISTER NUMBER USING PRIMARY GRAPH ALONE
NO
HAS A REGISTER NUMBER BEEN ALLOCATED TO ALL NODES?
48
YES
(TO 32)

FIG. 3